1 – Rekonstrukce citlivého dokumentu ze síťového provozu - Síťová analýza

Autor: **Pavel Minařík**

Zadání

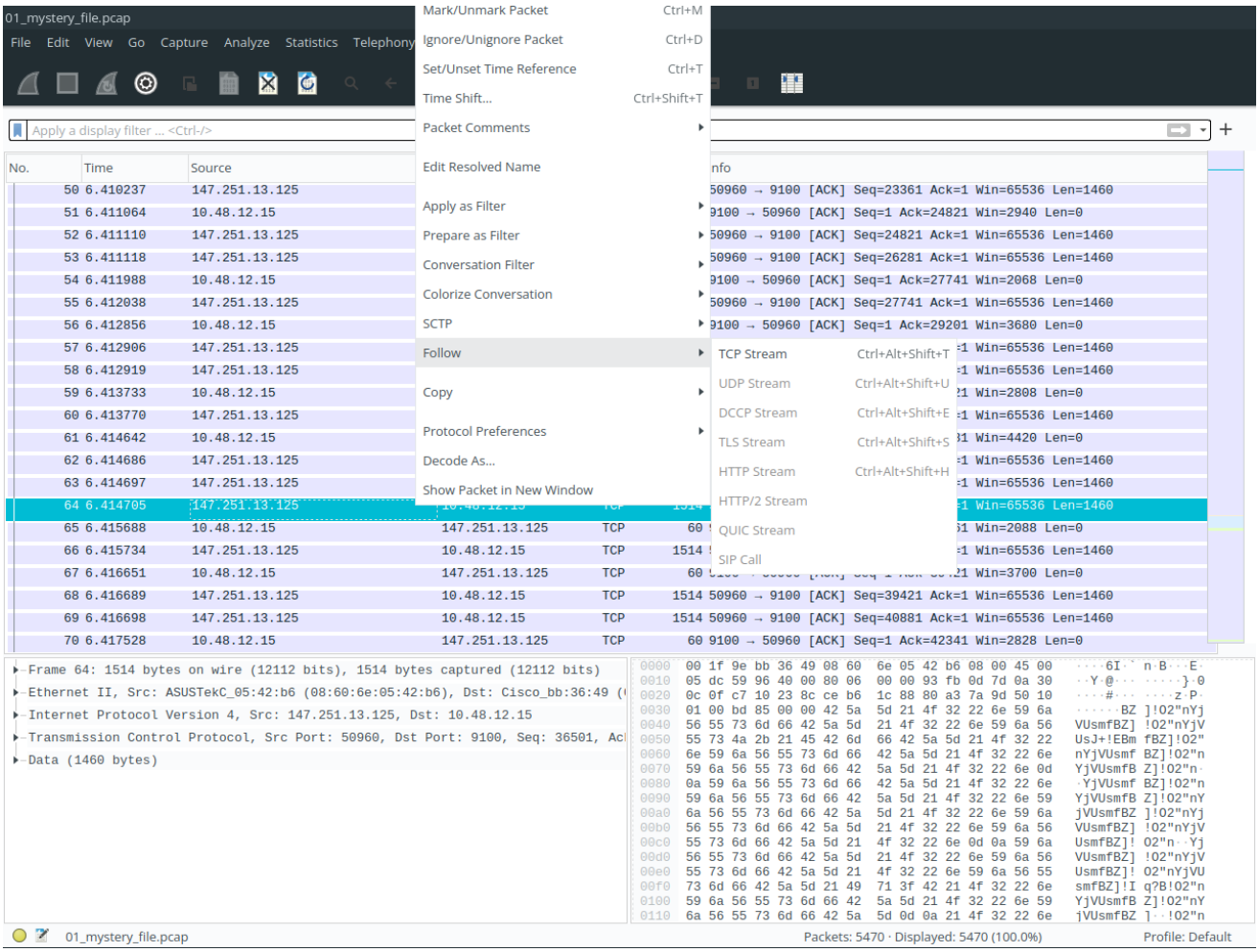
Existuje podezření na únik citlivých dat. Jako forenzní analytik jste získal přiložený záchyt provozu zaměstnance, který je za únik dat zřejmě zodpovědný. Dokážete zrekonstruovat obsah dokumentu, který unikl?

Příloha: `01_mystery_file.pcap`

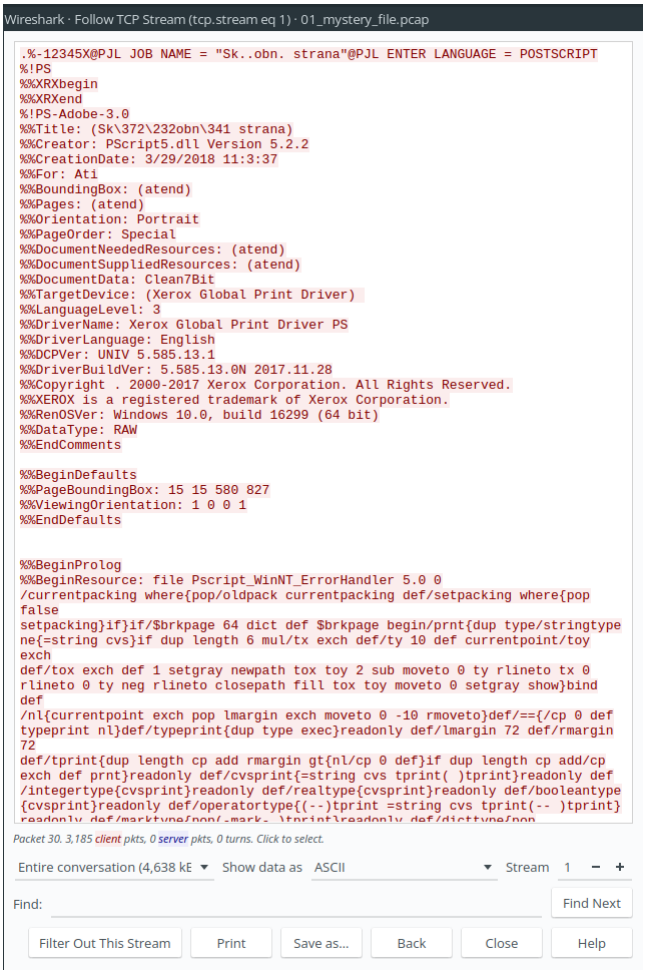
Řešení

Úloha je založená na **analýze síťového provozu**, a **extrakci neznámých dat**. Pomocí programu *Wireshark* ručně analyzujeme soubor pcap se síťovým provozem, zjistíme, že obsahuje komunikaci s tiskárnou na portu 9100 a extrahujeme z ní data. Tato data si následně prohlédneme a zjistíme, že se jedná o jazyk *PostScript* – předchůdce formátu PDF, který se hojně používá k odesílání dokumentů k tisku. Nakonec můžeme použít například Linuxový nástroj `gs` (GhostScript) na zobrazení obsahu souboru nebo `ps2pdf` na převedení do formátu PDF. Také je možné použít nějaký webový nástroj s podobnou funkcionalitou, například tento.

Nejprve otevřeme soubor ve Wiresharku. Všimneme si, že kromě pár packetů, které obsahují nesouvisející síťovou komunikaci se zde nachází pouze jeden TCP stream s cílovým portem 9100. Tento port používá mj. také protokol JetDirect, který používají tiskárny společnosti HP. Pro zobrazení přenesených dat v tomto streamu má Wireshark nástroj *Follow TCP Stream*, který vyvoláme kliknutím pravým tlačítkem na libovolný z paketů ze streamu:



Po kliknutí se nám zobrazí nové okno, ve kterém najdeme všechna přenesená data:



Odhadneme (buď vyhledáváním na internetu nebo z předchozí zkušenosti), že se jedná o soubor ve formátu PostScript. Pro další analýzu souboru se hodí soubor z Wiresharku exportovat. To je proces poněkud neintuitivní, je potřeba nejdříve přepnout políčko *Show data as* na *Raw* místo *ASCII* a poté **počkat několik vteřin**, než se obsah streamu znovu načte. Pak už stačí jen kliknout na *Save as…* a soubor s nějakým adekvátním názvem uložit.

Čeho si také ale musíme všimnout je, že co jsme získali, není ještě validní soubor PostScript, neboť první řádek tvoří hlavičku protokolu JetDirect (to můžeme také odhadnout díky tomu, že víme, že každý PostScriptový soubor začíná na `%!PS`). Odstranění prvního řádku můžeme provést v libovolném textovém editoru, např. `vim`, notepad.exe a podobně. Na linuxu lze také použít nástroj tail: `tail -n +2`.

Poté už libovolným způsobem převedeme soubor z PostScriptu do něčeho čitelného a získáme výsledný dokument.

Skúšobná strana tlačiarne systému Windows

Správne ste nainštalovali tlačiareň Xerox Global Print Driver PS do počítača DESKTOP-TC9IHI2.

VLASTNOSTI TLAČIARNE

Čas odoslania:	11:03:38
Dátum:	29.03.2018
Meno používateľa:	DESKTOP-TC9IHI2\Ati
Názov počítača:	DESKTOP-TC9IHI2
Názov tlačiarne:	Xerox Global Print Driver PS
Model tlačiarne:	Xerox Global Print Driver PS
Podpora pre farebnú tlač:	Áno
Názvy portov:	X_10_48_12_15
Formát údajov:	RAW
Názov zdieľanej tlačiarne:	Xerox Global Print Driver PS
Tlačový procesor:	winprint
Prostredie operačného systému:	Windows x64

VLASTNOSTI OVLÁDAČA TLAČIARNE

Názov ovládača:	Xerox Global Print Driver PS
Typ ovládača:	Typ 3 – Používateľský režim
Verzia ovládača:	5585.1300.0.0

DOPLNKOVÉ SÚBORY OVLÁDAČA TLAČE

C:\WINDOWS\system32\spool\DRIVERS\x64\3\xUNIVP0N.ini
C:\WINDOWS\system32\spool\DRIVERS\x64\3\xUNIVP0N.cfg
C:\WINDOWS\system32\spool\DRIVERS\x64\3\x2UNIV.ppd
C:\WINDOWS\system32\spool\DRIVERS\x64\3\x3up0N.dll
C:\WINDOWS\system32\spool\DRIVERS\x64\3\x3rps0N.dll
C:\WINDOWS\system32\spool\DRIVERS\x64\3\x3wfuv0N.dll
C:\WINDOWS\system32\spool\DRIVERS\x64\3\x3gui0N.dll
C:\WINDOWS\system32\spool\DRIVERS\x64\3\x3core0N.dll
C:\WINDOWS\system32\spool\DRIVERS\x64\3\x3util0N.dll
C:\WINDOWS\system32\spool\DRIVERS\x64\3\x3rnut0N.dll
C:\WINDOWS\system32\spool\DRIVERS\x64\3\x3coms0N.dll
C:\WINDOWS\system32\spool\DRIVERS\x64\3\x3jobt0N.exe
C:\WINDOWS\system32\spool\DRIVERS\x64\3\x3ptpc0N.dll
C:\WINDOWS\system32\spool\DRIVERS\x64\3\x3fput0N.dll
C:\WINDOWS\system32\spool\DRIVERS\x64\3\x3txt0N.cab
C:\WINDOWS\system32\spool\DRIVERS\x64\3\x3UNIV0N.cab
C:\WINDOWS\system32\spool\DRIVERS\x64\3\x3JAR0N.cab
C:\WINDOWS\system32\spool\DRIVERS\x64\3\x3fpb0N.exe
C:\WINDOWS\system32\spool\DRIVERS\x64\3\x3encr0N.dll
C:\WINDOWS\system32\spool\DRIVERS\x64\3\api-ms-win-core-file-l1-2-0.dll
C:\WINDOWS\system32\spool\DRIVERS\x64\3\api-ms-win-core-file-l2-1-0.dll
C:\WINDOWS\system32\spool\DRIVERS\x64\3\api-ms-win-core-localization-l1-2-0.dll
C:\WINDOWS\system32\spool\DRIVERS\x64\3\api-ms-win-core-processthreads-l1-1-1.dll
...

2 – Behaviorální analýza malware - Malware analýza

Autor: **Jan Pich**

Zadání

Kolegové z týmu bezpečnostního monitoringu identifkovali podezřelou aktivitu z jedné korporátní stanice. Jedním z indikátorů kompromitace, který je zaujal, je komunikace na IP adresu 1.248.122.240.

Jako člen forenzního týmu jste si vyžádali HASH všech souborů, se kterými uživatel v uplynulých 5 hodinách pracoval. HASHe, které tým dodal jsou uvedeny níže.

Identifikujte dle HASHe, který ze souborů je hledaný malware a odpovězte na soutěžní otázky.

- Soubor 1: 36C6F6214694EF64CC70F4127AC0CCEC668408A93825359D998FB31D24968D67
- Soubor 2: B5884FA3F05F88BBB617D08584930770C00BBCF675F2865A9161C2358829B605
- Soubor 3: 8d3f68b16f0710f858d8c1d2c699260e6f43161a5510abb0e7ba567bd72c965b
- Soubor 4: fbe196a583f84bb52db86ca1de63ddb6e2c8f11828f8632567e66ae2ddc5df22
- Soubor 5: AB0457BEFEBF51A9737E5CA9E46E34A3C37FFB5EF173CA5859E74D9EC76371C1

Otázky:

1. Identifikujte velikost souboru (v bytech, tzn. napište jen číslo, jednotky již nepište)
2. Malware modifikuje nastavení registru tak, aby se po spuštění operačního systému spouštěl soubor SAMPLE.EXE namísto jednoho systémového procesu. Identifikujte, o jaký systémový proces jde
3. Identifikujte celý název sekce (name) jenž je v HASH ovací funkci md5 rovna stringu f4ae31f7f77436e624b1667cc00af030
4. Identifikujte předěmet, který znázorňuje ikona daného malware souboru

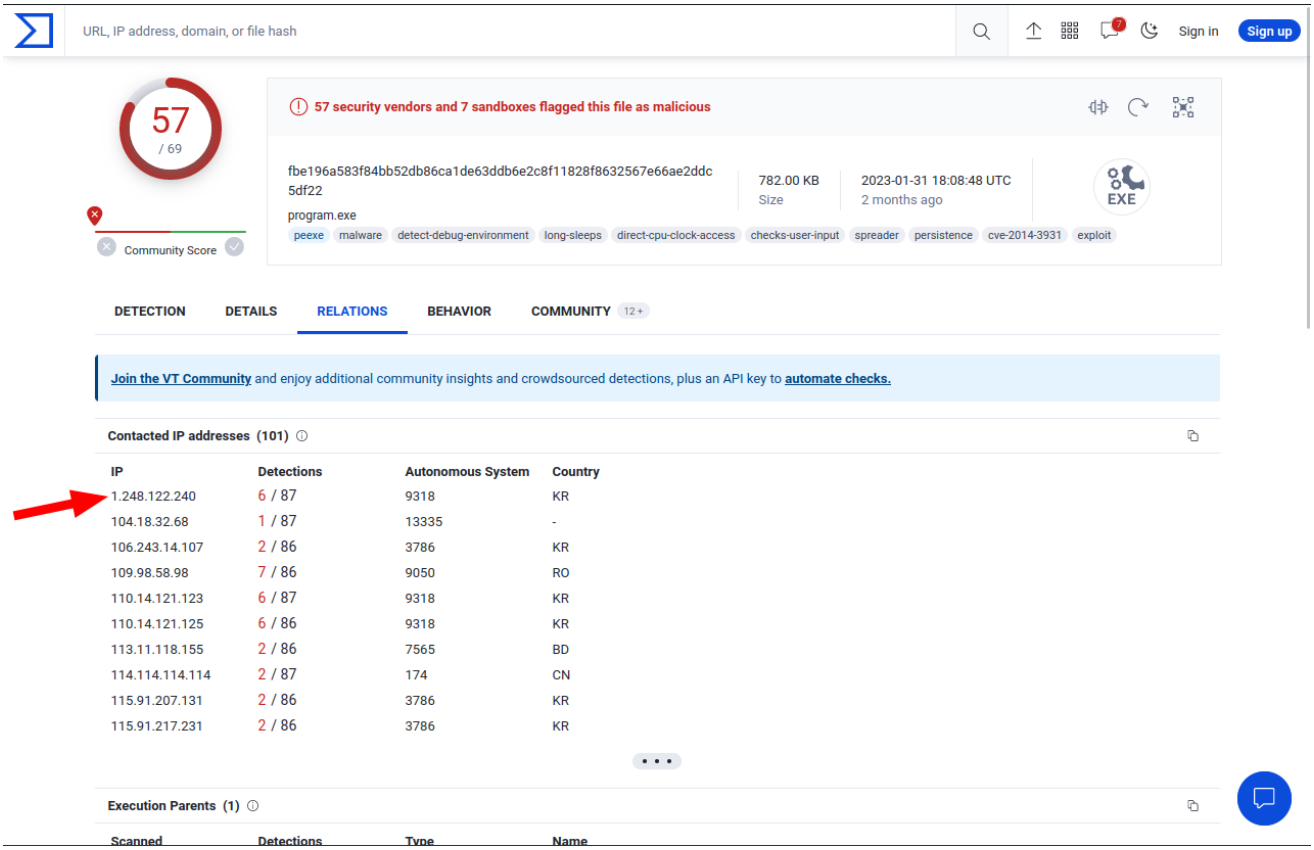
Řešení

(**Poznámka pro využití ve výuce:** Úloha zadaná tak jak je vyžaduje poměrně hodně „čtení myšlenek autora“, proto je vhodnější ji využít jako úlohu procvičovací, kdy pracuje třída společně s učitelem, případně upravit zadání tak, aby bylo trochu návodnější)

Úloha je, byť tak ze zadání nemusí vypadat, cvičení na práci se stránkou VirusTotal a jí podobných. Z toho důvodu také studentům není vůbec k dispozici samotný soubor s malwarem, mají k dispozici pouze hash, podle kterého informace o souboru na VirusTotalu najdou pomocí funkce *Search*.

Identifikace se kterým souborem pracujeme

V zadání jsme dostali pět různých hashů, musíme zjistit, který je ten, kterým se chceme zabývat. Když vyhledáme soubory 1, 2 a 5, zjistíme, že je VirusTotal neoznačuje jako škodlivé, proto je můžeme vyloučit. V úvodu zadání máme nápovědu, že hledaný malware komunikuje s IP adresou `1.248.122.240`, když se ve VirusTotalu podíváme u každého ze souborů do záložky *Relations* a sekce *Contacted IP addresses*, zjistíme, že právě soubor 4 s danou adresou komunikuje.



Otázka 1 – velikost souboru

Velikost souboru najdeme v záložce *Details*, sekci *Basic properties* pod klíčem *File size*. Zajímá nás velikost v bytech, tedy údaj uvedený v závorce – `800768`

Otázka 2 – nahrazený systémový proces

Zde budeme pracovat se záložkou *Behaviour*. Přijít zde na správné řešení není už tak přímočaré a nejspíš zabere hodně pročítání všech možných dat, která jsou na VirusTotalu dostupná, proto je vhodné studentům poskytnout případnou nápovědu.

Samotnou odpověď najdeme v sekci *Registry keys set*, kde třetí záznam `<HKCU>\Software\Microsoft\Windows\CurrentVersion\Run\SysHelper` obsahuje odkaz na zmíněný `SAMPLE.EXE` proto správná odpověď je `SysHelper`

Otázka 3 – název sekce

Zde je dlužno říci, že se jedná o sekce samotného .EXE souboru, což jsou jednotlivé části, které se samostatně kopírují do paměti a obsahují inicializované proměnné nebo samotný strojový kód programu. (**Poznámka pro využití ve výuce:** Terminologie je trochu matoucí, proto se hodí ji studentům nejprve osvětlit.)

Hashe jednotlivých sekcí najdeme v záložce *Details*, tentokrát trochu níže v sekci *Sections*. Všimneme si, že hash `f4ae ...` je uveden u sekce s názvem `.text`, což je správná odpověď.

Otázka 4 – ikona souboru

Tady uždnám VirusTotal nepomůže, neboť ikonu souboru nikde neukazuje. Budeme si muset pomoci jiným nástrojem, například službou AnyRun. Po kliknutí na tlačítko *Public tasks* můžeme procházet databázi souborů a pomocí původního hashe souboru (`fbe1 ...`) soubor v databázi najít. Nejjednodušší způsob je pak soubor stáhnout pomocí tlačítka *Get sample* (zde je potřeba vytvořit účet) a na ikonu souboru se podívat přímo.

POZOR: Stahujete živý soubor malwaru proto ho **za žádných okolností nespouštějte**. Ve výuce je pro bezpečnost možné tuto otázku přeskočit nebo ji jen předvést na počítači vyučujícího.

Ikona každopádně vypadá jako kalkulačka, což je i správná odpověď.

fbe196a583f84bb
52db86ca1de63d
db6e2c8f11828f...

4 – Munus Gyrari

Autor: **Eda Waber**

Zadání

Zjistěte, co dělá následující program a získejte vlajku.

```
from errno import ENXIO
import math
import random

EEZA = ""

BESKO_YKR = '}abcdefghijhlmnopqrstuvwxyz0123456789{'

random.seed(69)

JVLGU_UTXFC_M = random.sample(BESKO_YKR, len(BESKO_YKR))

JVLGU_UTXFC_M = "".join(JVLGU_UTXFC_M)

WLW = 420

def encrypt(mess, WLW):
  c = ""
  for i in mess:
    h = BESKO_YKR.find(i)
    h += WLW
    h %= len(BESKO_YKR)
    c += JVLGU_UTXFC_M[h]

  return c

c = "4qdmf9veoezotbvonxdx4}hedzvae}zabvzz4j"
```

Řešení

Nejdříve kód prozkoumáme, odstraníme řádky, které nic nedělají a přidáme komentáře, co která řádka dělá.

```
import random

abeceda = '}abcdefghijhlmnopqrstuvwxyz0123456789{'

random.seed(69)    # "Zasazení semínka" náhodného generátoru.
# Všechny volání funkcí knihovny `random` vrátí pokaždé stejnou náhodnou hodnotu

klic = random.sample(abeceda, len(abeceda)) # Vytvoří klíč náhodným přeházením znaků v šifrovací abecedě

klic = "".join(klic) # Ze seznamu znaků vytvoří string

ceasar_offset = 420

def encrypt(plaintext, ceasar_offset):
    ciphertext = ""
    for znak in plaintext:
        poradi = abeceda.find(znak)  # Najde pořadí znaku v abecedě
```

```
        poradi += ceasar_offset      # Posune pořadí o 420 (jako caesarova šifra)
        poradi %= len(abeceda)       # Omezí pořadí na délku abecedy
        ciphertext += klic[poradi]   # Vybere z klíče znak v daném pořadí
(monoalfabetická substituce)
    return ciphertext

ciphertext = "4qdmf9veoezotbvonxdx4}hedzvae}zabvzz4j"
```

Z kódu je zřetelné, že se jedná o nějakou šifrovací funkci a na posledním řádku se nachází nějaká zašifrovaná zpráva. Víme také, že klíč je konstantní a můžeme jej získat zavoláním řádek, které jej generují, samostatně:

```
Python 3.10.10 (main, Mar 01 2023, 21:10:14) [GCC] on linux
Type "help", "copyright", "credits" or "license" for more information.
>>> import random
>>> abeceda = '}abcdefghijhlmnopqrstuvwxyz0123456789{'
>>> random.seed(69)
>>> klic = random.sample(abeceda, len(abeceda))
>>> klic = "".join(klic)
>>>
>>> klic
'bfjdvtzn4mr3p8qw271l{si6cu5yhxa}oh9g0e'
```

Samotnou dešifrovací funkci pak můžeme poté prostě připsat na konec souboru ze zadání a celý soubor spustit:

```
abeceda = '}abcdefghijhlmnopqrstuvwxyz0123456789{'
klic = 'bfjdvtzn4mr3p8qw271l{si6cu5yhxa}oh9g0e' # Klíč můžeme napsat do kódu rovnou,
generuje se

ceasar_offset = 2  # Modulo na řádce 14 vždy číslo ořízne mod 38, můžeme nahradit

def encrypt(plaintext, ceasar_offset):
    ciphertext = ""
    for znak in plaintext:
        poradi = abeceda.find(znak)  # Najde pořadí znaku v abecedě
        poradi += ceasar_offset      # Posune pořadí o 2 (jako caesarova šifra)
        poradi %= len(abeceda)       # Omezí pořadí na délku abecedy
        ciphertext += klic[poradi]   # Vybere z klíče znak v daném pořadí
(monoalfabetická substituce)
    return ciphertext

def decrypt(ciphertext, ceasar_offset):
    plaintext = ""
    for znak in ciphertext:
        poradi = klic.find(znak)
        poradi += len(abeceda) - ceasar_offset # Zaručíme, že se modulo nikdy nedostane
pod nulu
        poradi %= len(abeceda)
        plaintext += abeceda[poradi]
    return plaintext

ciphertext = "4qdmf9veoezotbvonxdx4}hedzvae}zabvzz4j"
print(decrypt(ciphertext, ceasar_offset))
```

Tento program vypíše `flag{5b838d3c9b3e0a0f2z8adb182d19bddf}`, což je výsledná vlajka.

7 – Hledání skrytého hashe

Autor: **Jan Pich**

Zadání

Jste tajným agentem na velmi rizikové misi v zahraničí. Za žádnou cenu nesmí vaše identita nijakým způsobem být odhalena, ale i nadále potřebujete komunikovat se svou zemí tak, aby to nebylo odhaleno. Zvolili jste proto pro aktuální dobu jednu z nejběžnějších forem komunikace – pomocí obrázků. V obrázcích se skrývají tajné šifry, ke kterým byste se měli dostat jenom vy. Jeden z takových obrázků jste právě od spojenců obdržel. Soubor obrázku má obsahovat zahashované heslo, které budete potřebovat pro vykonání své mise.

Příloha: `07_tajnyobrazek.jpg`

Řešení

Soubor s obrázkem ve skutečnosti tvoří dva soubory slepené k sobě. Toho si můžeme všimnout po otevření souboru v nějakém HEX editoru – soubory JPEG obvykle končí na `FFD9`, soubory ZIP naopak obvykle začínají na `504B0304` (první dva byty jsou v ASCII `PK` a jsou dobrým poznávacím znamením ZIP souborů, neboť jsou na první pohled vidět). Obě dvě tyto znamení v souboru najdeme hned za sebou, proto můžeme soubory od sebe buď „odlepit“ ručně, nebo použít program binwalk, který s parametrem `-e` automaticky všechny skryté soubory extrahuje.

Odlepený soubor ZIP obsahuje jeden textový soubor s neznámým HEXovým textem. Jedná se o hash `0262176d7e7b9d7ef838290618f2890b`, po vložení do Crackstation zjistíme, že se jedná o MD5 hash slova `kockopes`.

8 - Kyblíková - Cloud

Autor: **Mikuláš Hrdlička**

Zadání

Někde na webu http://bucketlist.ctf.hackinglab.cz/ se nachází vlajka. Najdete ji?

Řešení

Jedná se o webovou úlohu, proto je na místě nějaký průzkum, čtení HTML zdrojového kódu, proklikávání stránky, kontrola známých souborů (`/robots.txt` ...) atd., ale to je mimo rozsah tohoto bulletinu.

Během průzkumu zjistíme, že doména `bucketlist.ctf.hackinglab.cz.` má CNAME záznam na `bucketlist.ctf.hackinglab.cz.s3-website.eu-central-1.amazonaws.com.`:

```
; <<>> DiG 9.18.13 <<>> bucketlist.ctf.hackinglab.cz
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 4864
;; flags: qr rd ra; QUERY: 1, ANSWER: 3, AUTHORITY: 0, ADDITIONAL: 1

;; OPT PSEUDOSECTION:
; EDNS: version: 0, flags:; udp: 512
;; QUESTION SECTION:
;bucketlist.ctf.hackinglab.cz.  IN  A

;; ANSWER SECTION:
bucketlist.ctf.hackinglab.cz. 74 IN CNAME   bucketlist.ctf.hackinglab.cz.s3-website.eu-central-1.amazonaws.com.
bucketlist.ctf.hackinglab.cz.s3-website.eu-central-1.amazonaws.com. 300 IN CNAME s3-website.eu-central-1.amazonaws.com.
s3-website.eu-central-1.amazonaws.com. 5 IN A   52.219.168.44

;; Query time: 48 msec
;; SERVER: ::1#53(::1) (UDP)
;; WHEN: Sun Apr 09 16:45:36 CEST 2023
;; MSG SIZE  rcvd: 167
```

Jedná se tedy o web hostovaný pomocí služby Amazon S3, což je služba určená k ukládání souborů, které mohou být dostupné po HTTP. Stránka proto není dynamická, jedná se pouze o předgenerované soubory, které S3 na požadavek vrací.

Pokud nahradíme `s3-website` na `s3` v adrese, která se nachází v CNAME záznamu, dostaneme přímý přístup k API S3. Za normálních okolností by to nešlo, ale úložiště je pro tuto stránku špatně nakonfigurované proto to jde. Když otevřeme http://bucketlist.ctf.hackinglab.cz.s3.eu-central-1.amazonaws.com/ (pozor na `s3` místo `s3-website`), dostaneme výpis všech souborů v úložišti ve formátu XML. I bez nutnosti nějakého prohlížeče se můžeme dočíst, že je v XML zmíněn soubor `5244cecd-8218-4ef9-ac1c-43c250b9cf39/flag.txt`, který vypadá velmi zajímavě. Když tuto cestu přidáme na konec původní adresy stránky, dostaneme vlajku:

```
$ curl http://bucketlist.ctf.hackinglab.cz/5244cecd-8218-4ef9-ac1c-43c250b9cf39/flag.txt
FLAG{AlwaysCheckAWSPermissions-8452}
```

Alternativním řešením je použití programu `aws-cli`, což je univerzální nástroj pro práci se službami od Amazonu:

```
$ aws s3 ls s3://bucketlist.ctf.hackinglab.cz --no-sign-request
                           PRE 5244cecd-8218-4ef9-ac1c-43c250b9cf39/
                           PRE css/
```

```
                           PRE img/
                           PRE js/
                           PRE mail/
                           PRE scss/
                           PRE vendor/
2020-02-14 10:16:53        208 .browserslistrc
2020-02-14 10:16:53         12 .gitignore
2020-02-14 10:16:53        199 .travis.yml
2020-02-14 10:16:53       1093 LICENSE
2020-02-14 10:16:53       4309 README.md
2020-02-14 10:16:53       3563 gulpfile.js
2022-11-11 10:14:40      16118 index.html
2020-02-14 10:16:53     266039 package-lock.json
2020-02-14 10:16:53       1393 package.json
$ aws s3 ls s3://bucketlist.ctf.hackinglab.cz/5244cecd-8218-4ef9-ac1c-43c250b9cf39/ --
no-sign-request
2022-11-11 10:16:32         36 flag.txt
$ aws s3 cp s3://bucketlist.ctf.hackinglab.cz/5244cecd-8218-4ef9-ac1c-
43c250b9cf39/flag.txt - --no-sign-request
FLAG{AlwaysCheckAWSPermissions-8452}
```

---

13 - Volby - Datová analýza

Autor: **Michal Kopřiva**

Zadání

Ve dnech 23. a 24. září 2022 proběhly volby do obecních zastupitelstev. Z důvodu nalezení vhodného krycího jména pro pracovníky zpravodajských služeb najdi:

- Nejčastější příjmení, které kandidovalo
- Kolikrát to bylo - MD5 hash tohoto čísla je FLAG1
- Nejčastější křestní jméno
- Kolikrát to bylo - MD5 hash tohoto čísla je FLAG2

Řešení

(**Poznámka pro využití ve výuce:** Pokud úlohu zadáte jako úkol, je vhodné požadovat i postup řešení a případně uznávat řešení, která z nějakého důvodu nevytvoří správný výsledek (ať už kvůli využití jiné datové sady nebo nějaké lehké chybě v měření), i když je postup myšlenkově správný a třeba se liší o nízké jednotky jmen. Konkrétní nejčastější jména by měla vyjít téměř vždy stejná)

Na https://www.volby.cz/opendata/kv2022/kv2022_opendata_seznam.htm najdeme mnoho zajímavých sad s otevřenými daty. Nás konkrétně bude zajímat Registr kandidátů. Jedná se o CSV soubor který můžeme zpracovat více způsoby, například pomocí jednoduchého Python skriptu:

```
import csv

surnames = {}
names = {}

with open("./kvrk.csv") as csvfile:
    reader = csv.reader(csvfile)
    next(reader)
    for row in reader:
        names[row[7]] = names.get(row[7],0)+1
        surnames[row[8]] = surnames.get(row[8],0)+1

print(sorted(surnames.items(), key=lambda x: (-x[1], x[0]))[0])
print(sorted(names.items(), key=lambda x: (-x[1], x[0]))[0])
```

Nebo pomocí kontingenční tabulky v LibreOffice Calcu.

Autorské řešení, které platilo během soutěže je:

```
 892 Novák
9146 Petr
```

V době tvorby tohoto dokumentu byla ale data aktualizována a výsledek je:

```
 902 Novák
9176 Petr
```

MD5 hashe získáme například pomocí nástroje `md5sum` nebo libovolného onlinového nástroje.

```
$ echo "902" | md5sum
3d9dc0cd2ee9b2c19407a88c7c1fb4b6
```

---

17 - Macrohard without macro

Autor: **Eda Waber**

Zadání

Zjistěte kam komunikuje soubor. Soubor může být antiviry označován jako škodlivý avšak žádnou neplechu nedělá.

Heslo k .zip souboru je: infected

Vlajka je ve formátu flag{}

Příloha: `17-Macrohard-withoout-macro.zip`

Řešení

Jedním z možných řešení, které může přinést zajímavý vhled do problematiky, je analýza pomocí automatizovaných nástrojů jako je VirusTotal. Zjistíme například, že dokument používá nedávno objevenou zranitelnost ve Wordu (CVE-2021-40444).

Jde to ale také přímočařeji. Soubory, které používá Microsoft Office (`.docx`, `.xlsx`, `.pptx`…) jsou ve skutečnosti jen přejmenované soubory `.zip`, tak se pojďme podívat dovnitř.

Uvnitř zipu najdeme následující složkovou strukturu:

```
.
├── [Content_Types].xml
├── docProps
│   ├── app.xml
│   └── core.xml
├── _rels
└── word
    ├── document.xml
    ├── fontTable.xml
    ├── _rels
    │   └── document.xml.rels
    ├── settings.xml
    ├── styles.xml
    ├── theme
    │   └── theme1.xml
    └── webSettings.xml
```

Tu si můžeme celou proklikat, ale vzhledem k tomu, že dokument je kromě škodlivého kódu prázdný, nenajdeme tam mnoho zajímavého. Můžete si zkusit takto proklikat jiný `.docx` soubor, můžete tam najít například obrázky nebo celý obsah dokumentu zakódovaný v XML v souboru `word/document.xml`.

Po proklikání celého škodlivého souboru zjistíme, že soubor `word/_rels/document.xml.rels` obsahuje URL, která tam rozhodně nepatří a tou je `https://nakit.cz/kyb3rs0u73z/`. Během soutěže při otevření v prohlížeči tato adresa vrátila vlajku, ale to už není funkční, proto jako odpověď stačí tato URL.

Následující úlohy jsou trochu pokročilejší a popis řešení předpokládá částečnou znalost problematiky.

5 – Odhal API klíč v mobilní aplikaci

Autor: **Matěj Sychra**

Zadání

Pomocí dekompileru zanalyzujte přiložený balíček TestApplication.apk a najděte klíč ve formátu UUID, které má aplikace vestavěný.

Přílohy: `05,06-TestApplication-v1.apk`, `05,06-TestApplication-v2.apk`

Pozn.: Tato úloha se během druhého kola změnila, proto uvádíme obě verze v příloze, stejně tak postupy řešení obou verzí.

Řešení (v1)

`.apk` balíček dekompilujeme například pomocí dekompilátoru jadx (který je dostupný i online). Víme, že vlajka má být ve formátu UUID, proto stačí pomocí nástroje `grep` a regulárního výrazu pro tento formát vlajku mezi zdrojovým kódem najít:

```
1  grep -rPn '[0-9a-f]{8}-[0-9a-f]{4}-[0-9a-f]{4}-[0-9a-f]{4}-[0-9a-f]{12}' .
2  ./sources/cz/corpus/dva/logindata/LoginData.java:10:        return "cf32f2f8-592e-11ed-
   934d-c7a5b9711d27";
```

Případně můžeme využít `jadx-gui`, což je klikací varianta nástroje `jadx` a zdrojový kód proklikat ručně, případně prohledat pomocí *Navigation → Text search* stejně, jako pomocí `grep` u výše.

Řešení (v2)

Opět využijeme `jadx` (případně `jadx-gui`) pro dekompilaci, tektokrát ale UUID které hledáme není ve formátu UUID, jak zadání napovídá, ale proměnná, ve které je UUID uloženo se jmenuje `UUID`. Když opět použijeme `grep` pro vyhledání tohoto klíčového slova, najdeme více výsledků, ze kterých vybereme ten relevantní:

```
1  ./sources/cz/corpus/dva/ui/login/LoginActivity.java:76:    private final String UUID =
   "687474703a2f2f7468696e782e636c6f75643a333333332f";
```

6 – Sledování mobilní komunikace

Autor: **Matěj Sychra**

Zadání

Pomocí sledování síťové komunikace zjisti, na kterou URL odesílá mobilní aplikace polohu zařízení.

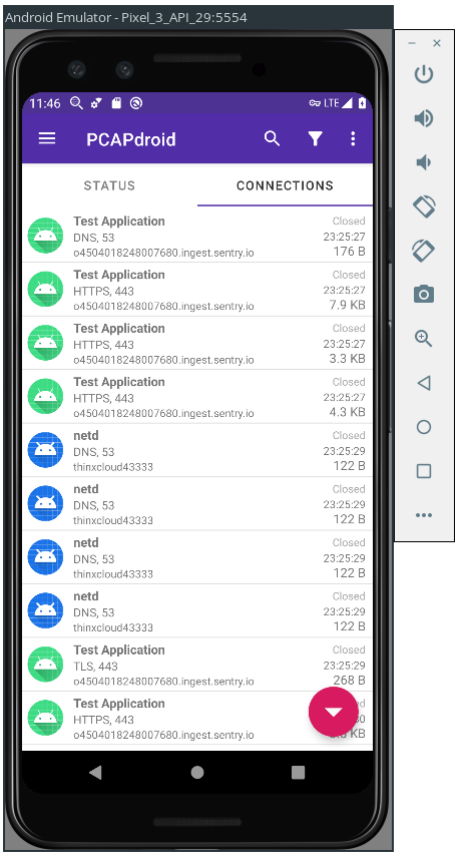
Pozn.: Tato úloha se během druhého kola změnila, proto uvádím obě verze v příloze, stejně tak postupy řešení obou verzí.

Přílohy: `05,06-TestApplication-v1.apk`, `05,06-TestApplication-v2.apk`

Řešení (v1)

(**Poznámka pro využití ve výuce:** Zadání úlohy není úplně jednoznačné, proto se nehodí k přímému zadání například jako hodnocený domácí úkol, opět spíše pro vyzkoušení si práci se zmíněnými nástroji.)

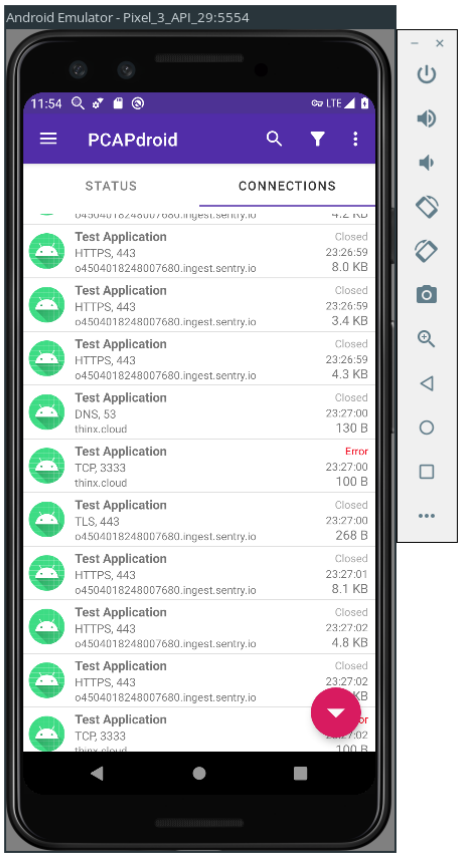
Budeme potřebovat nějakou aplikaci pro záchyt síťové komunikace na Androidu a ideálně také Android Emulátor. Emulátor použijeme ten, který je přibalený s Android Studiem, na záchyt komunikace použijeme PCAPdroid.



Ve výpisu vidíme, že aplikace kontaktuje sentry.io, kam ale posílá pouze logy svého běhu nikoliv polohu. Také se pokouší kontaktovat adresu http://thinxcloud43333, což ale není validní DNS adresa, proto se ve výpisu zobrazí pouze DNS request od procesu netd, nikoliv od aplikace samotné. http://thinxcloud43333 je každopádně hledaná odpověď.

Řešení (v2)

Opět použijeme téměř stejný postup jako v předchozí verzi, jen adresa už alespoň v DNS existuje, takže se ukáže ve výpisu spojení normálně: http://thinx.cloud:3333


Android Emulator - Pixel_3_API_29:5554
11:54
PCAPdroid
STATUS
CONNECTIONS
Test Application
HTTPS, 443
o4504018248007680.ingest.sentry.io
Closed
23:26:59
8.0 KB
Test Application
HTTPS, 443
o4504018248007680.ingest.sentry.io
Closed
23:26:59
3.4 KB
Test Application
HTTPS, 443
o4504018248007680.ingest.sentry.io
Closed
23:26:59
4.3 KB
Test Application
DNS, 53
thinx.cloud
Closed
23:27:00
130 B
Test Application
TCP, 3333
thinx.cloud
Error
23:27:00
100 B
Test Application
TLS, 443
o4504018248007680.ingest.sentry.io
Closed
23:27:00
268 B
Test Application
HTTPS, 443
o4504018248007680.ingest.sentry.io
Closed
23:27:01
8.1 KB
Test Application
HTTPS, 443
o4504018248007680.ingest.sentry.io
Closed
23:27:02
4.8 KB
Test Application
HTTPS, 443
o4504018248007680.ingest.sentry.io
Closed
23:27:02
Test Application
TCP, 3333


CYBER SECURITY
COMPETITION
CZECHIA
www.kybersoutez.cz